CS 4495 Computer Vision

Features 2 – SIFT descriptor

Aaron Bobick
School of Interactive Computing

Administrivia

- PS 3: Out – due Oct 12th.

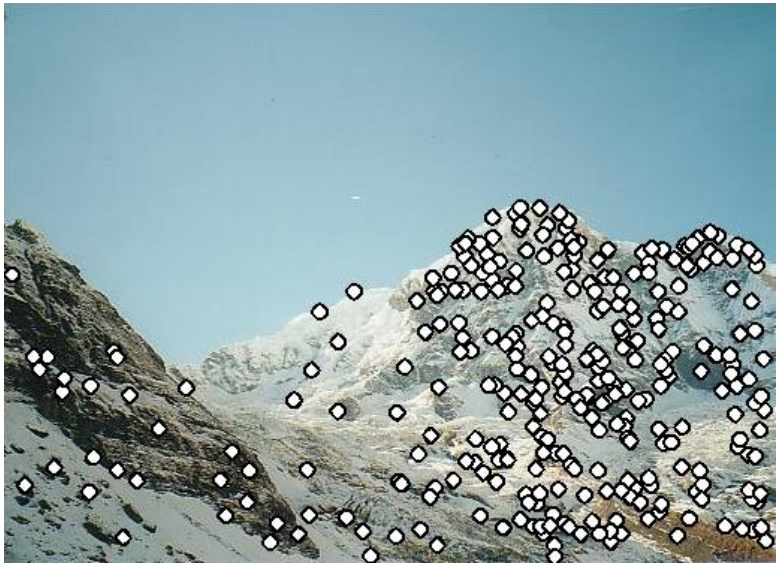
- Features recap:
  - Goal is to find corresponding locations in two images.
  - Last time: find locations that can be accurately located and likely to be found in both images even if photometric or slight geometric changes.

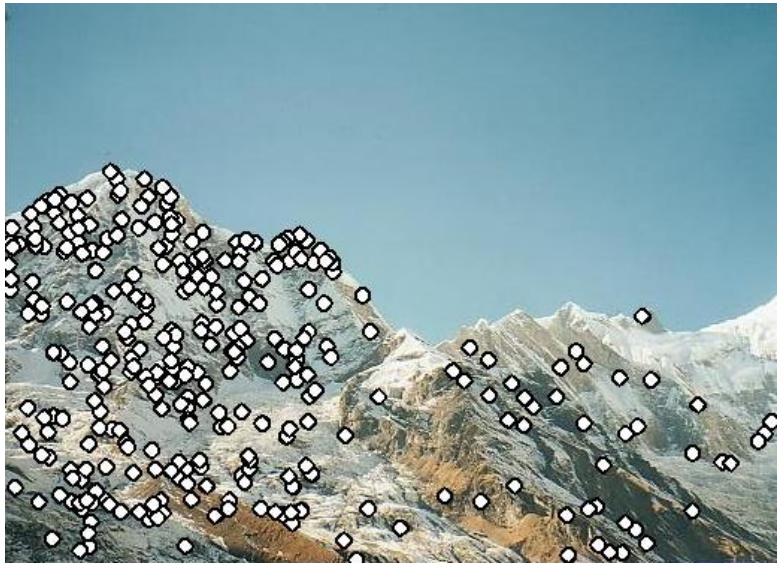
  - This time– find possible (likely?) correspondences between points
  - Next: which of the guessed, plausible correspondences are correct

- Today's part on matching done really well in Szeliski section 4.1

Matching with Features

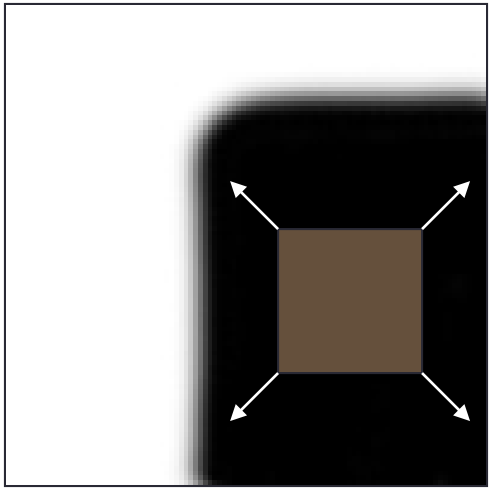
- Detect feature points in both images



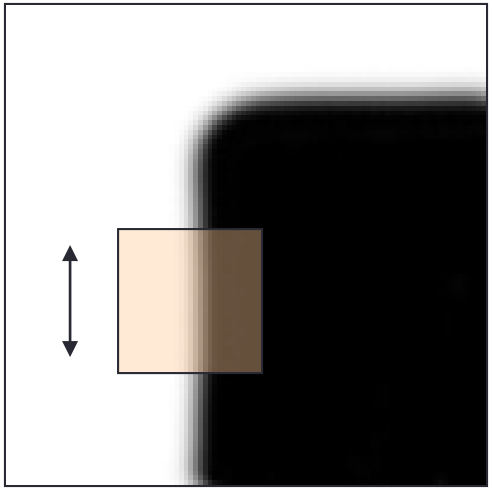


Corner Detection: Basic Idea

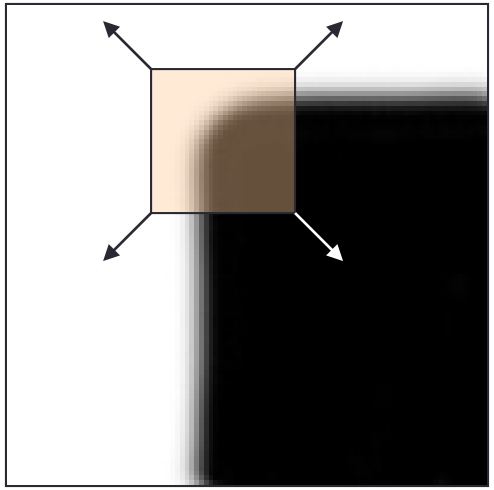
- We should easily recognize the point by looking through a small window
- Shifting a window in any direction should give a large change in intensity



"flat" region: no change in all directions



"edge": no change along the edge direction

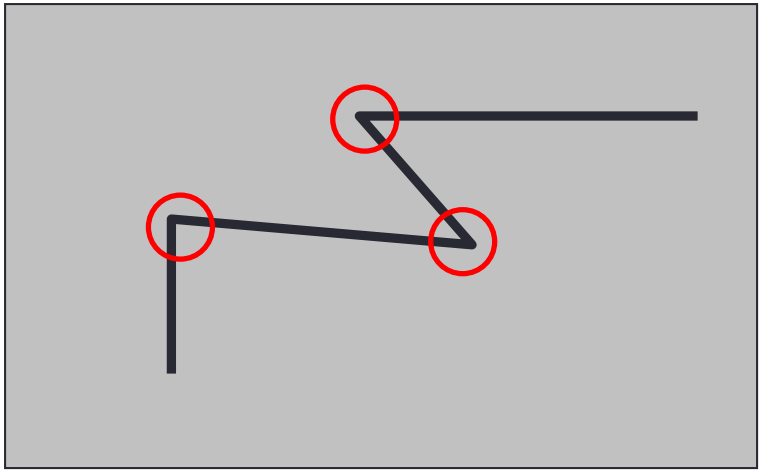


"corner": significant change in all directions with small shift

Source: A. Efros

An introductory example:

Harris corner detector



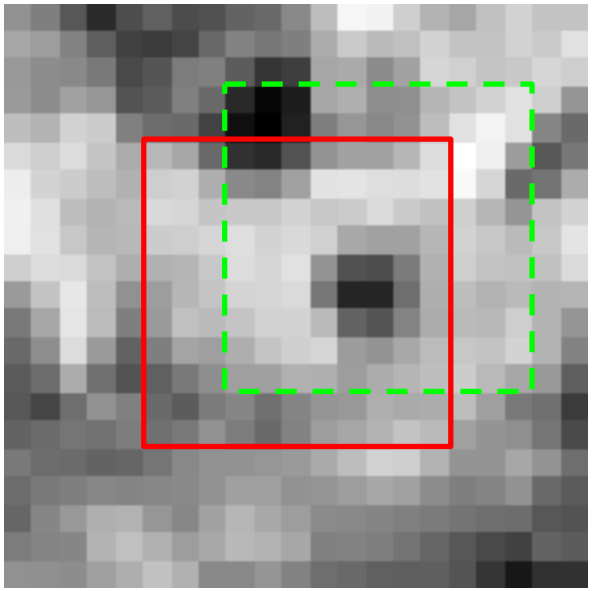
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

Corner Detection: Mathematics

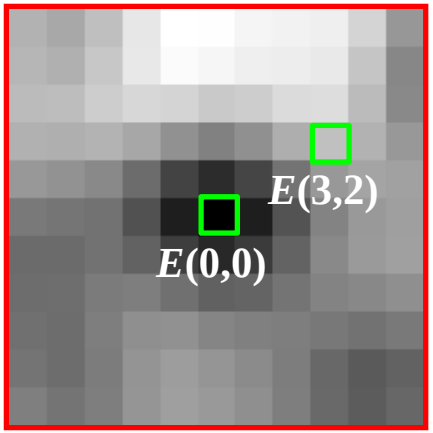
Change in appearance for the shift [*u,v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

$I(x, y)$



$E(u, v)$



Corner Detection: Mathematics

Change in appearance for the shift [*u,v*]:

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of *E*(*u*,*v*) about (0,0) (local quadratic approximation for small *u*,*v*):

$$(1\mathrm{D}): F(\delta x) \approx F(0) + \delta x \cdot \frac{dF(0)}{dx} + \frac{1}{2}\delta x^2 \cdot \frac{d^2F(0)}{dx^2}$$

$$E(u,v) \approx E(0,0) + [u \;\; v]\begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2}[u \;\; v]\begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Corner Detection: Mathematics

$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a *second moment matrix* computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Each product is a rank 1 2x2

Without weight

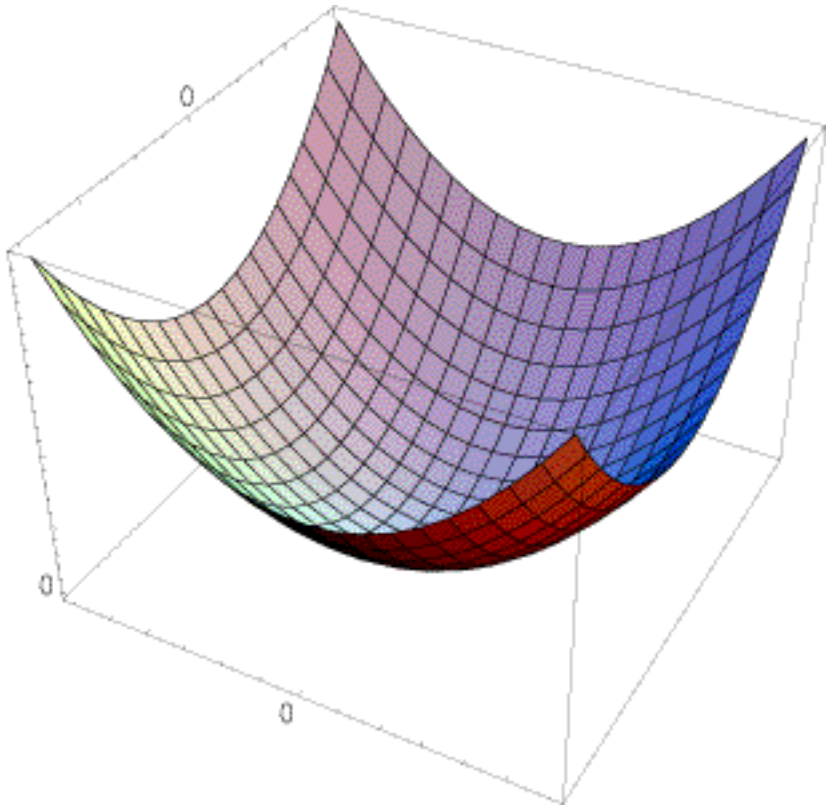
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

Interpreting the second moment matrix

Thus the surface $E(u,v)$ is locally approximated by a quadratic form (no linear term).

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

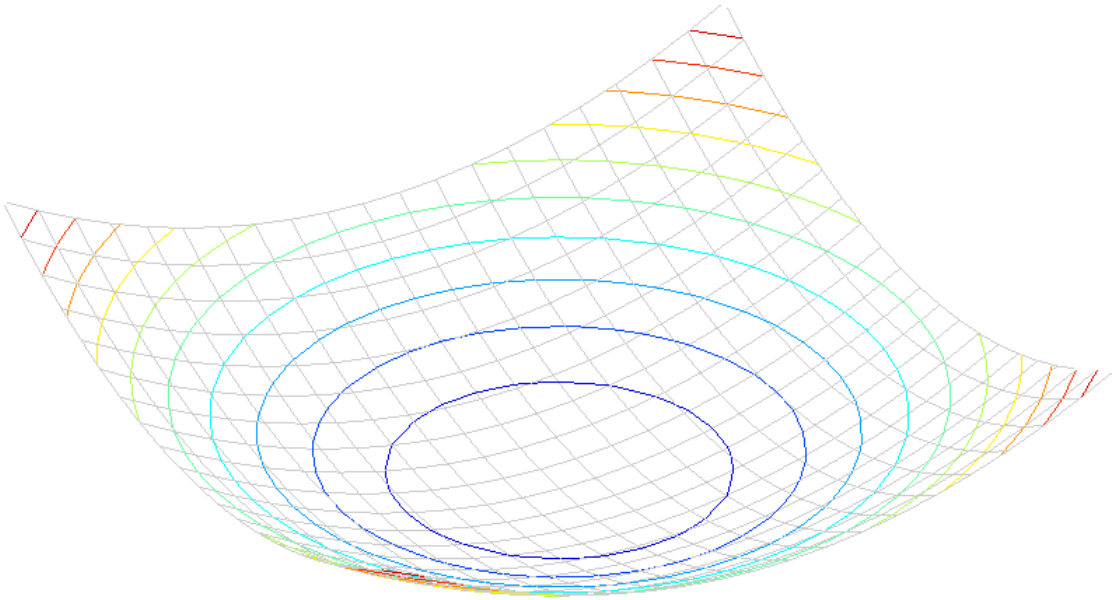


Interpreting the second moment matrix

Consider a constant "slice" of $E(u, v)$: $[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

$$\left(\sum I_x^2\right)u^2 + \left(2\sum I_x I_y\right)uv + \left(\sum I_y^2\right)v^2 = k$$

This is the equation of an ellipse in u,v. If $\Sigma I_x I_y$ is zero, then aligned with the u,v axes



Interpreting the second moment matrix

If rotated to align the ellipse with the axes then:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & 0 \\ 0 & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

The bigger λ_1 the faster the surface goes up in *u;* the bigger λ_2 the faster the surface goes up in *v.*

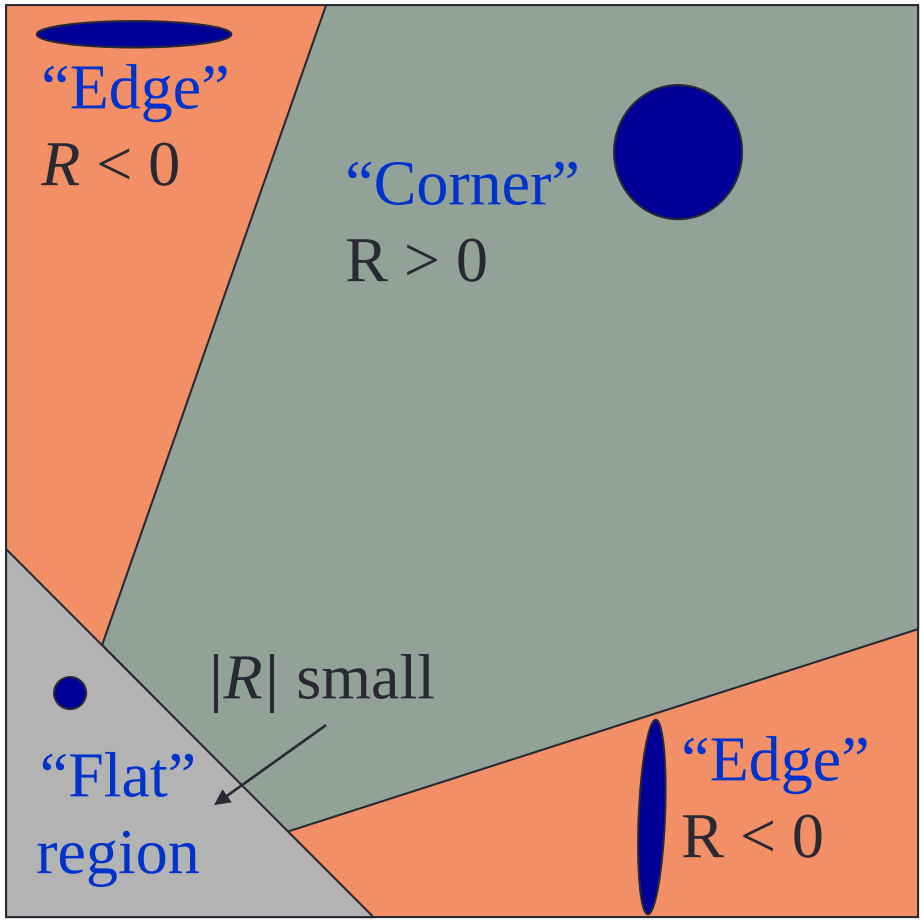
If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

Harris corner response function

$$R = \det(M) - \alpha \,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

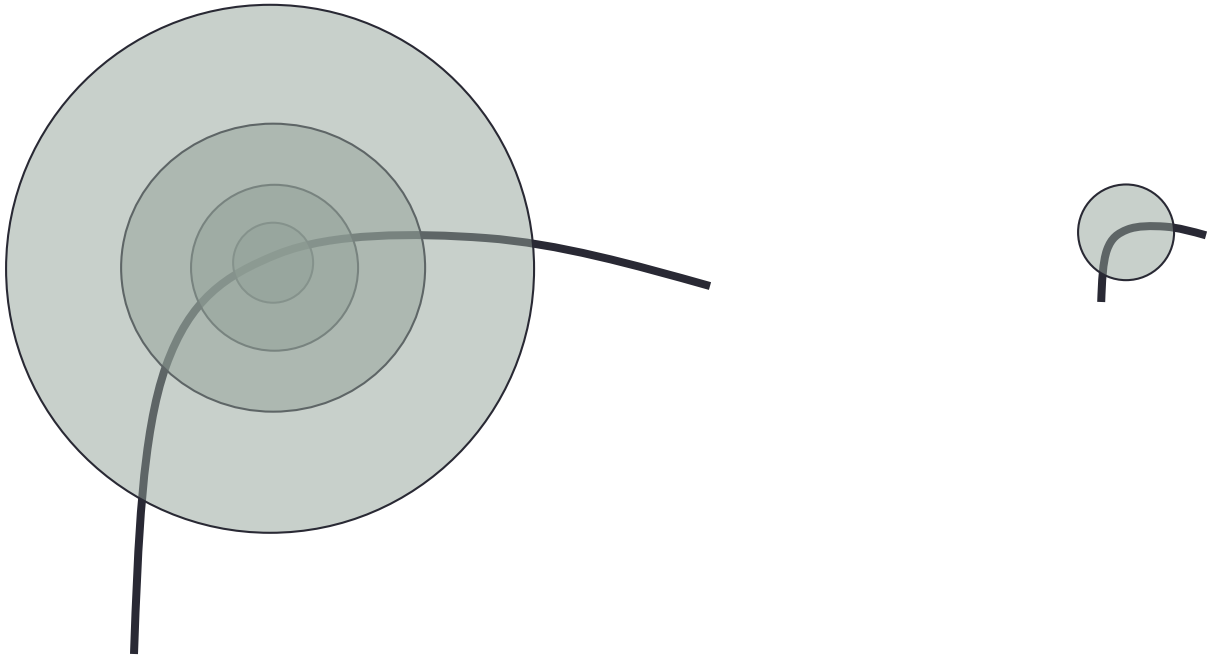
α: constant (0.04 to 0.06)

- R depends only on eigenvalues of M, but don't compute them (no sqrt, so really fast!
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



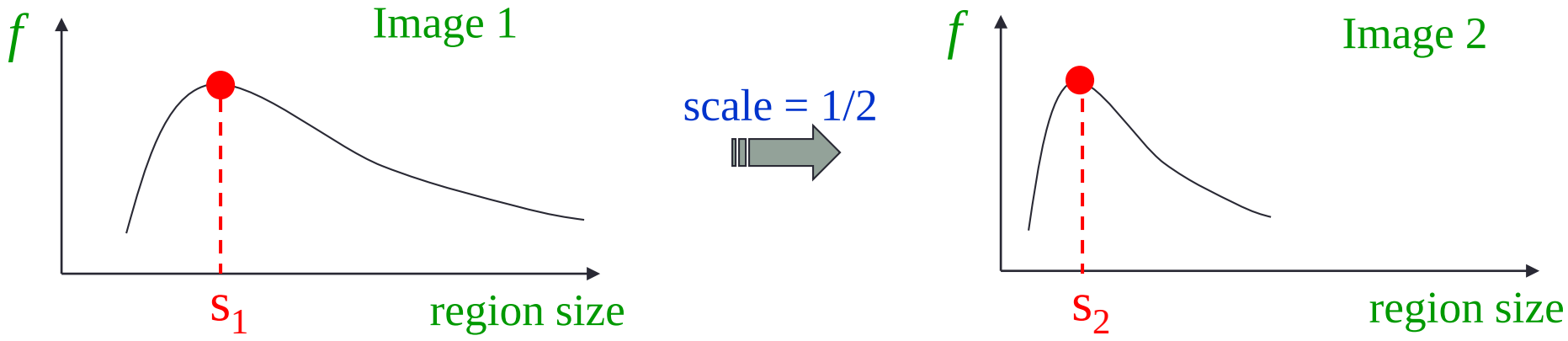
Scale Invariant Detection

- Common approach:

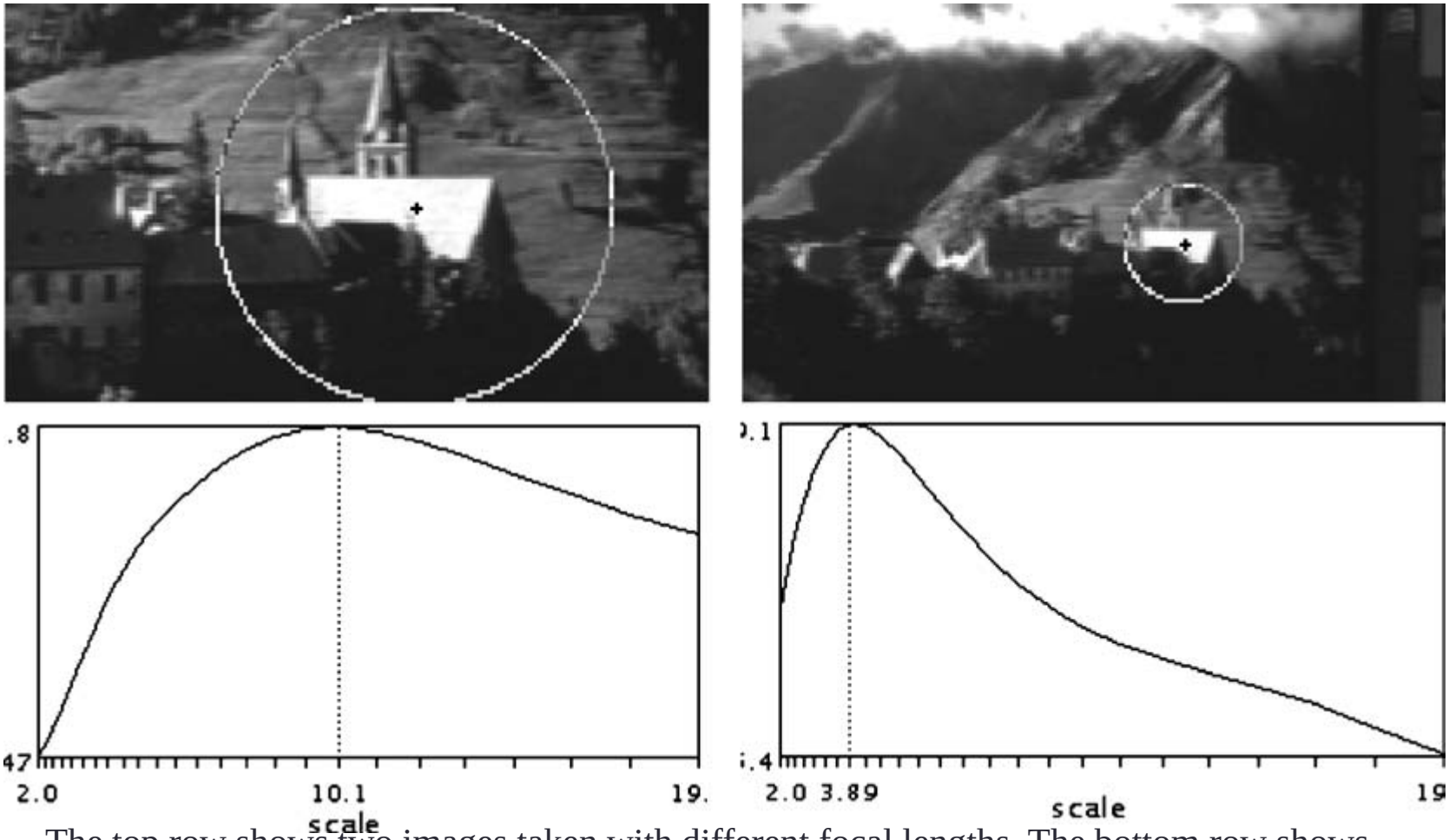
  Take a local maximum of this function

  Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



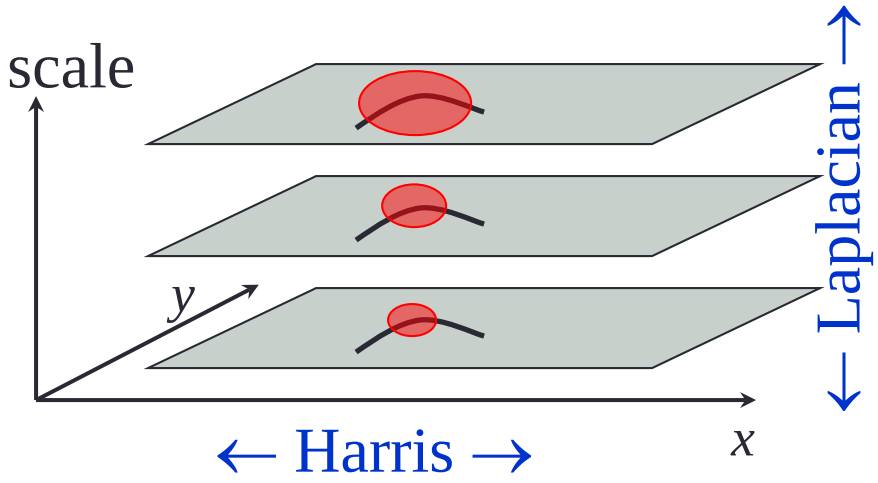
Scale sensitive response



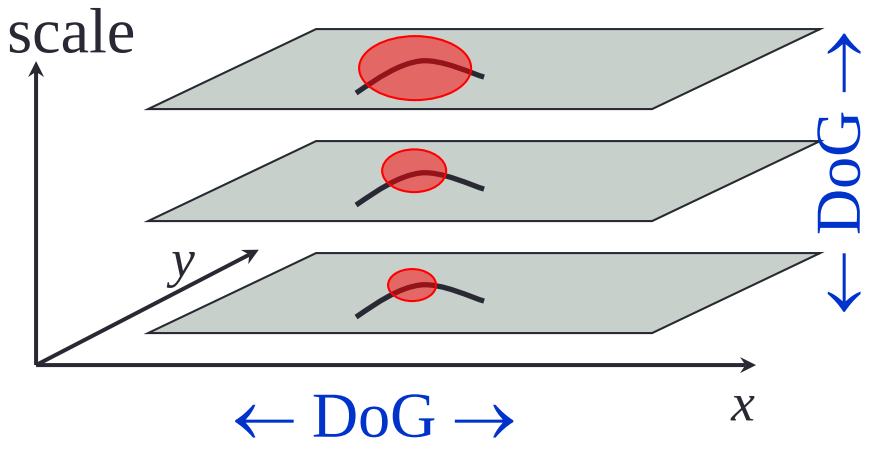
The top row shows two images taken with different focal lengths. The bottom row shows the response over scales of the normalized LoG . The ratio of scales corresponds to the scale factor (2.5) between the two images.

Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



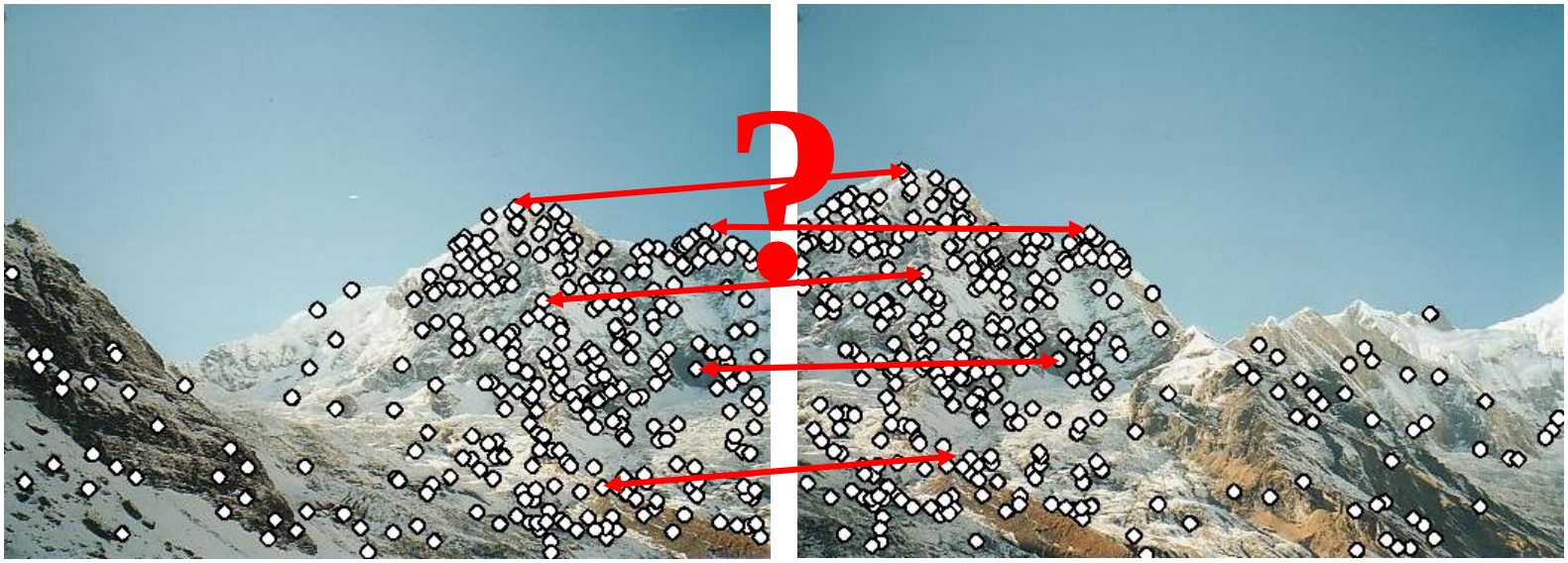
- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

Point Descriptors

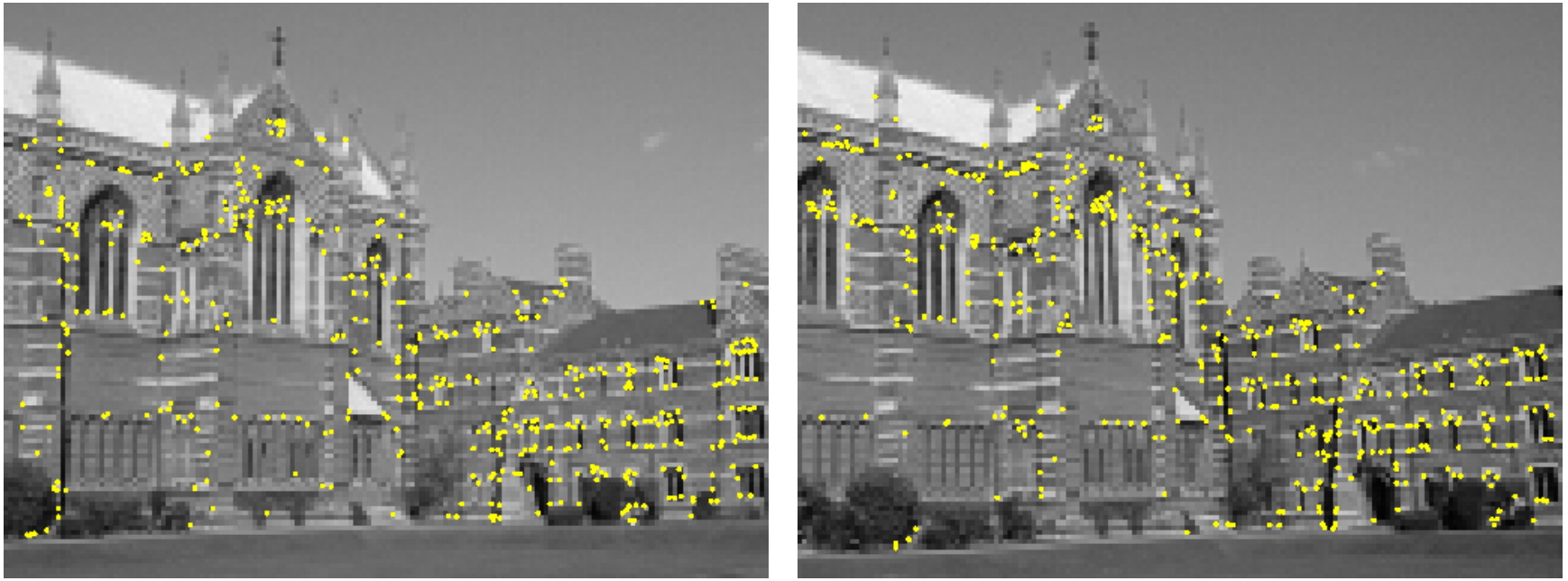
- We know how to detect points
- Next question: How to match them?



Point descriptor should be:

1. Invariant
2. Distinctive

Harris detector



Interest points extracted with Harris (~ 500 points)

Simple solution?

- Harris gives good detection – and we also know the scale.
- Why not just use correlation to check the match of the window around the feature in image1 with every feature in image 2?

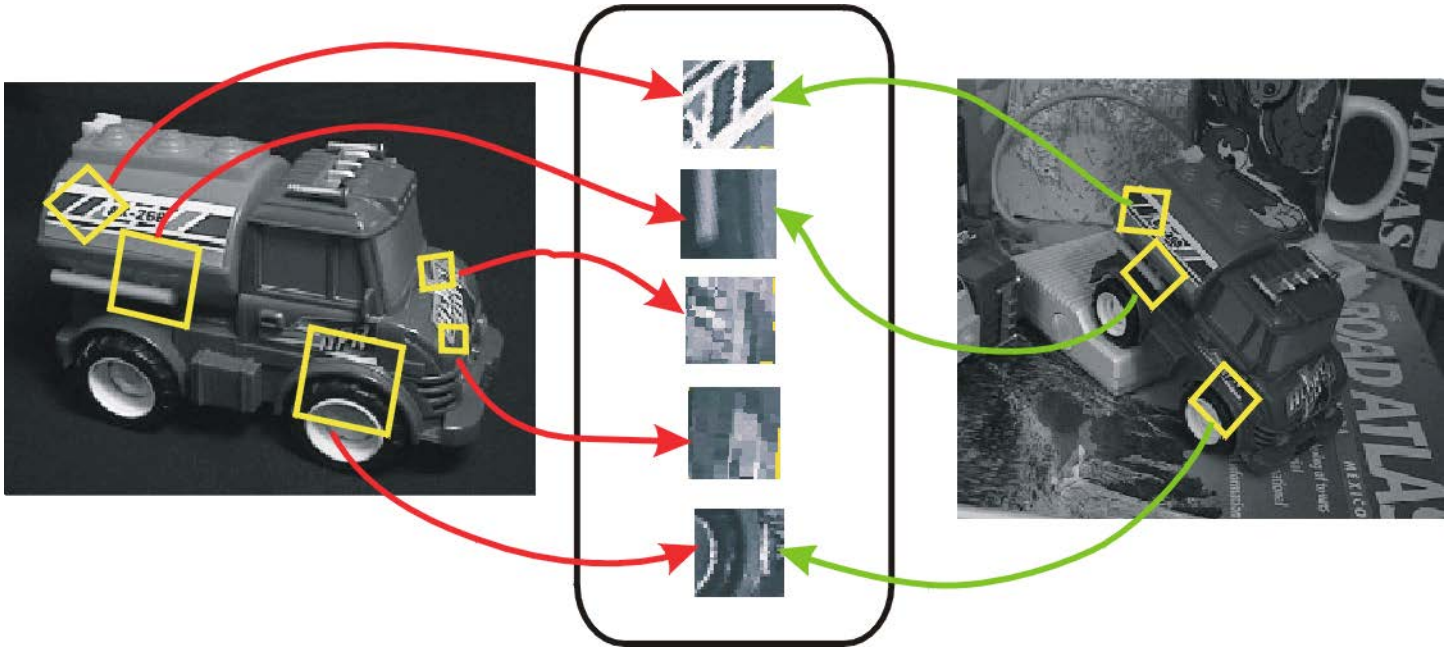
- Main reasons:
  1. Correlation is not rotation invariant - why do we want this?
  2. Correlation is sensitive to photometric changes.
  3. Normalized correlation is sensitive to non-linear photometric changes and even slight geometric ones.
  4. Could be slow.

SIFT: Motivation

- The Harris operator is not invariant to scale and correlation is not invariant to rotation.
- For better image matching, Lowe's goal was to develop an interest operator – a *detecto*r – that is invariant to scale and rotation.
- Also, Lowe aimed to create a descriptor that was robust to the variations corresponding to typical viewing conditions. The descriptor is the most-used part of SIFT.

Idea of SIFT

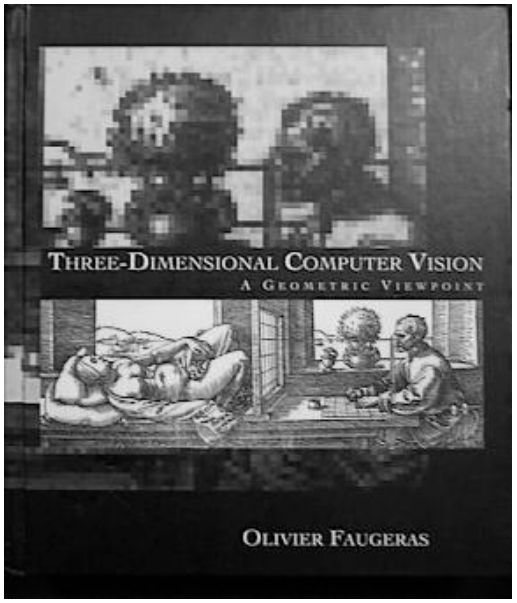
- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

Another version of the problem…

Want to find



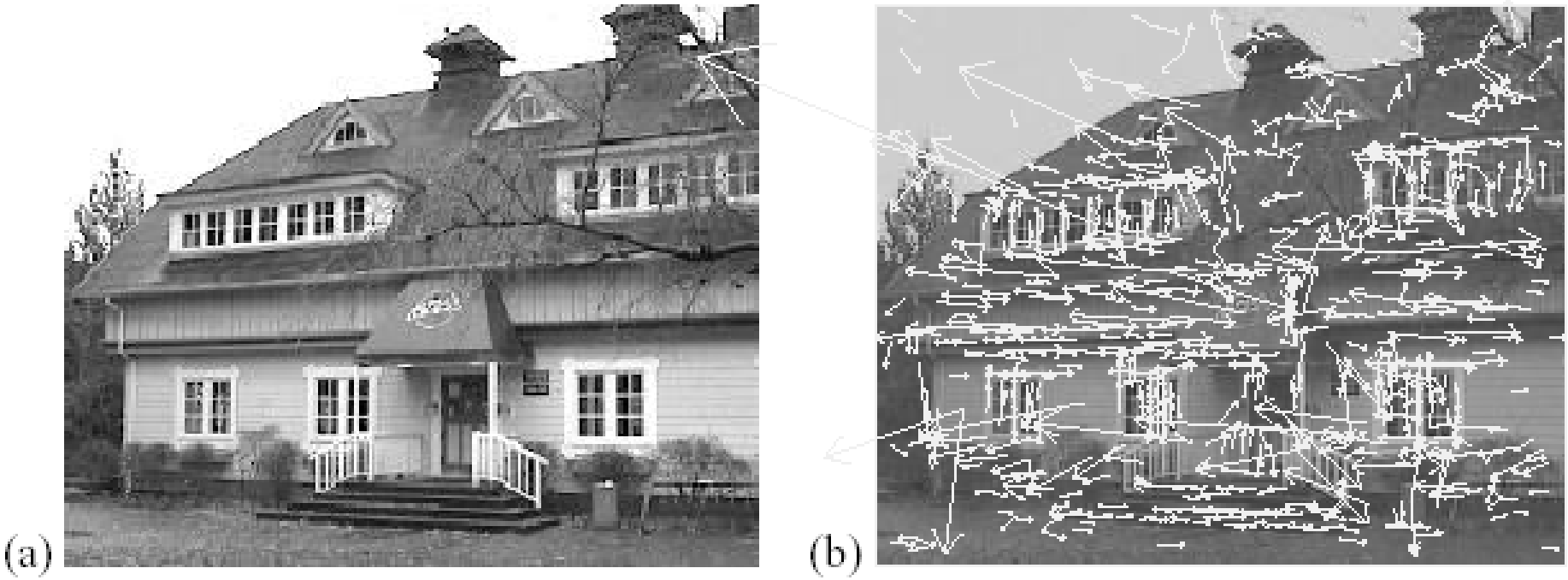
… in here

Overall Procedure at a High Level

- Scale-space extrema detection
  - Search over multiple scales and image locations
- Keypoint localization
  - Define a model to determine location and scale. Select keypoints based on a measure of stability.
- Orientation assignment
  - Compute best orientation(s) for each keypoint region.
- Keypoint description
  - Use local image gradients at selected scale and rotation
  - to describe each keypoint region.

Use Harris-Laplace or other method

Example of keypoint detection



(a) 233x189 image
(b) 832 DOG extrema

Overall Procedure at a High Level

1. Scale-space extrema detection

   Search over multiple scales and image locations

2. Keypoint localization

   Define a model to determine location and scale.
   Select keypoints based on a measure of stability.

3. Orientation assignment

   Compute best orientation(s) for each keypoint region.

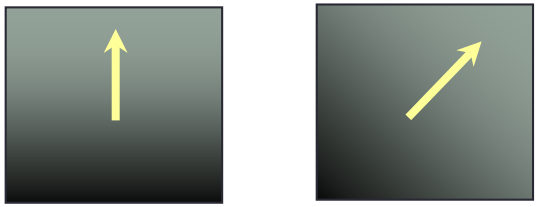
4. Keypoint description

   Use local image gradients at selected scale and rotation to describe each keypoint region.

Descriptors Invariant to Rotation

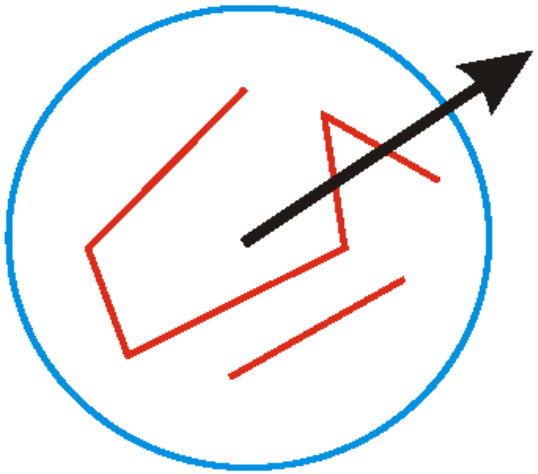
- Find local orientation

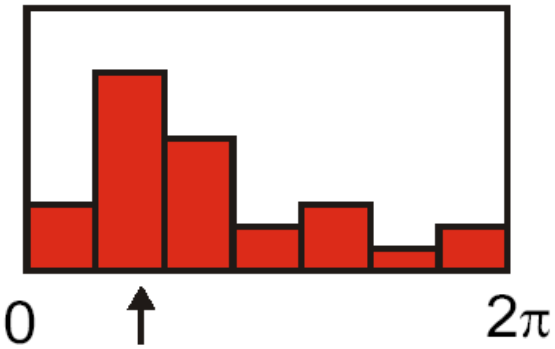
Dominant direction of gradient



- Compute image derivatives relative to this orientation

3. Orientation assignment





- Create histogram of local gradient directions at selected scale – 36 bins
- Assign canonical orientation at peak of smoothed histogram
- Each keypoint now specifies stable 2D coordinates (x, y, scale, orientation) – invariant to those.

If a few major orientations, use 'em all...how?

4. Keypoint Descriptors

- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

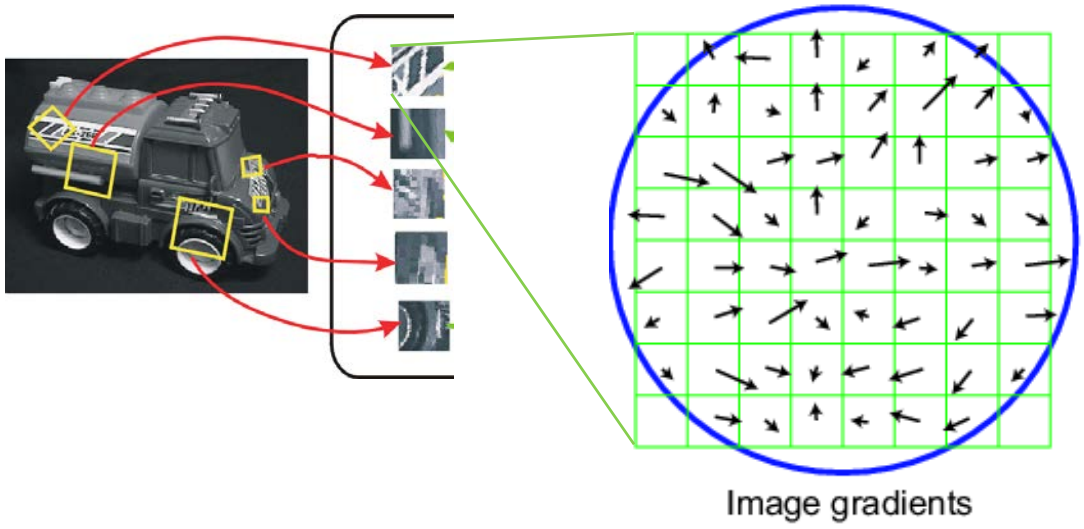
But first… normalization

- Rotate the window to standard orientation

- Scale the window size based on the scale at which the point was found.

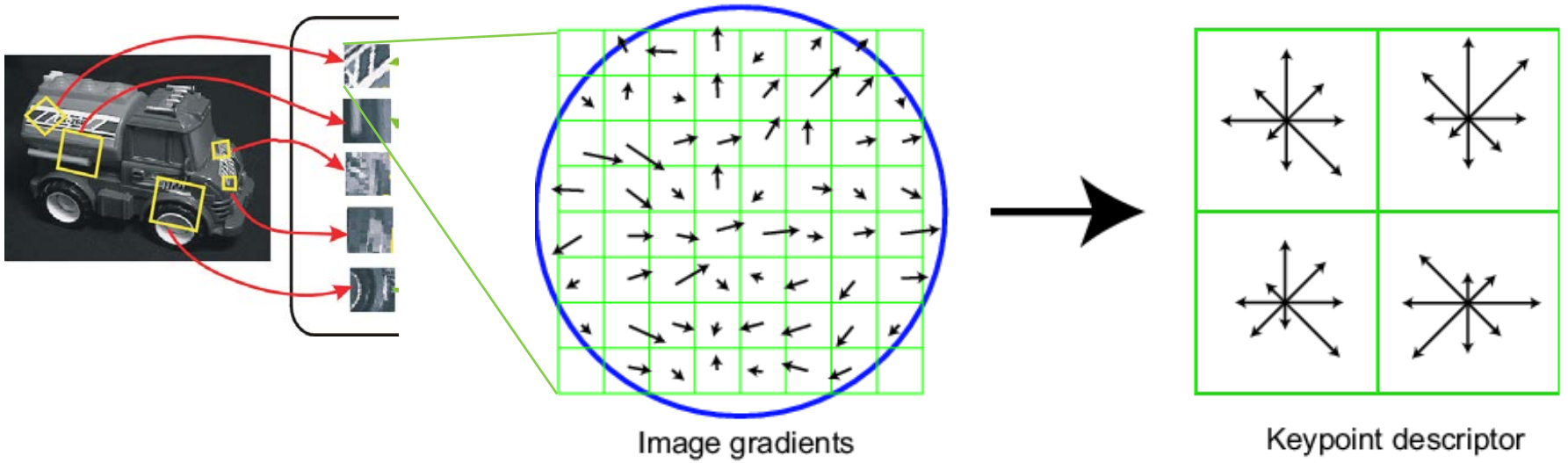
SIFT vector formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)



SIFT vector formation

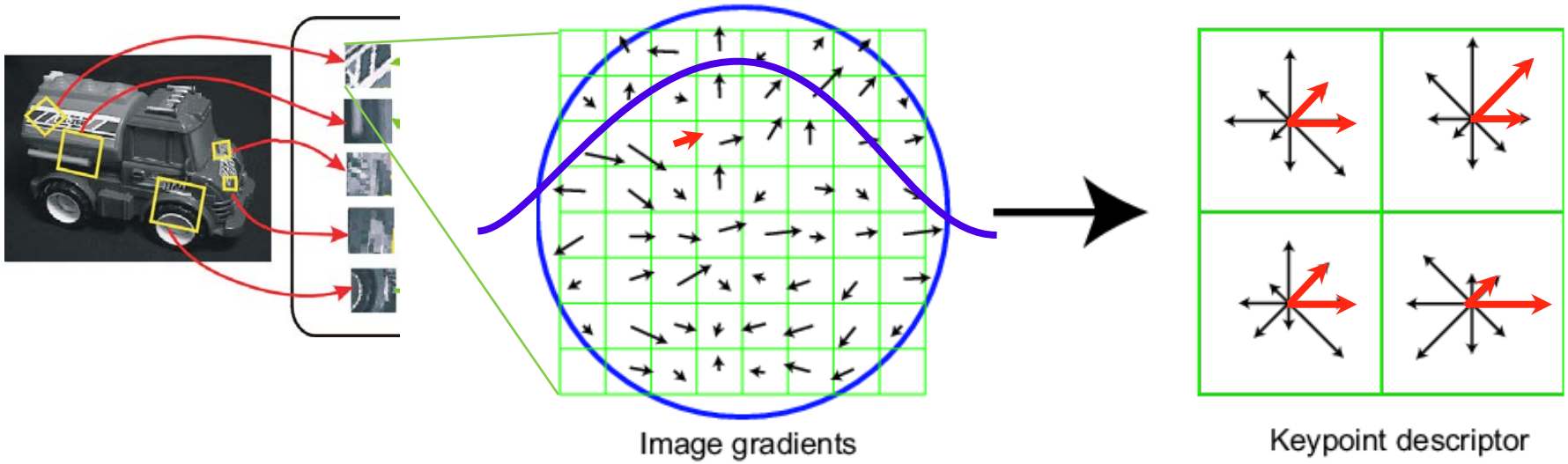
- 4x4 array of gradient orientation histograms over 4x4 pixels
  - not really histogram, weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.



showing only 2x2 here but is 4x4

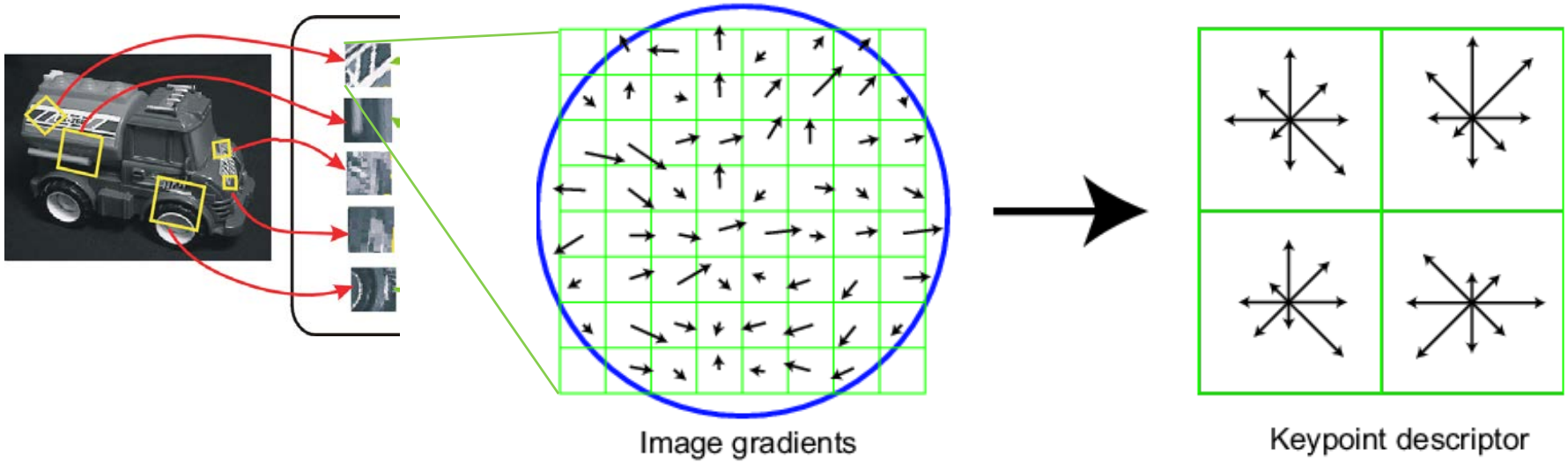
Ensure smoothness

- Gaussian weight
- “Trilinear” interpolation
  - a given gradient contributes to 8 bins: 4 in space times 2 in orientation



Reduce effect of illumination

- 128-dim vector normalized to magnitude 1.0
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after rotation normalization, clamp gradients >0.2



Evaluating the SIFT descriptors

- Database images were subjected to rotation, scaling, affine stretch, brightness and contrast changes, and added noise. Feature point detectors and descriptors were compared before and after the distortions, and evaluated for:

- Sensitivity to number of histogram orientations and subregions.
- Stability to noise.
- Stability to affine change.
- Feature distinctiveness

Sensitivity to number of histogram orientations and subregions,

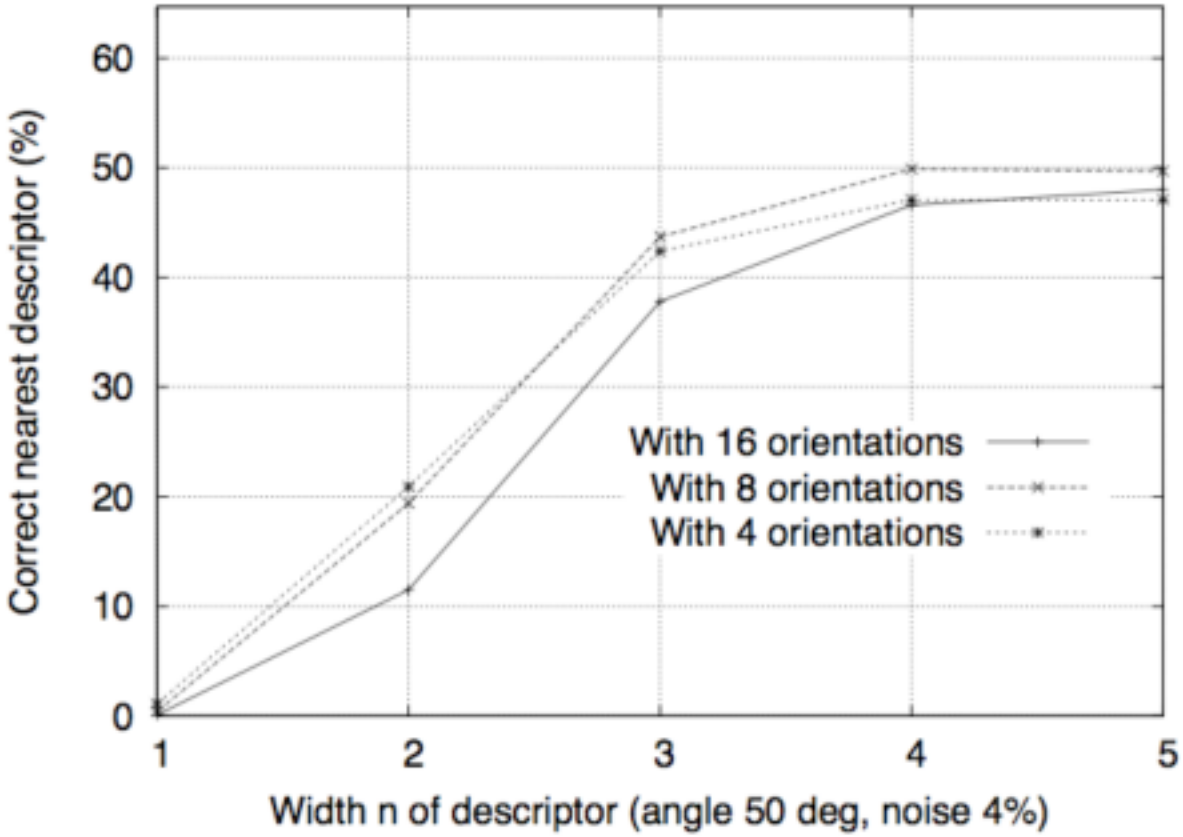
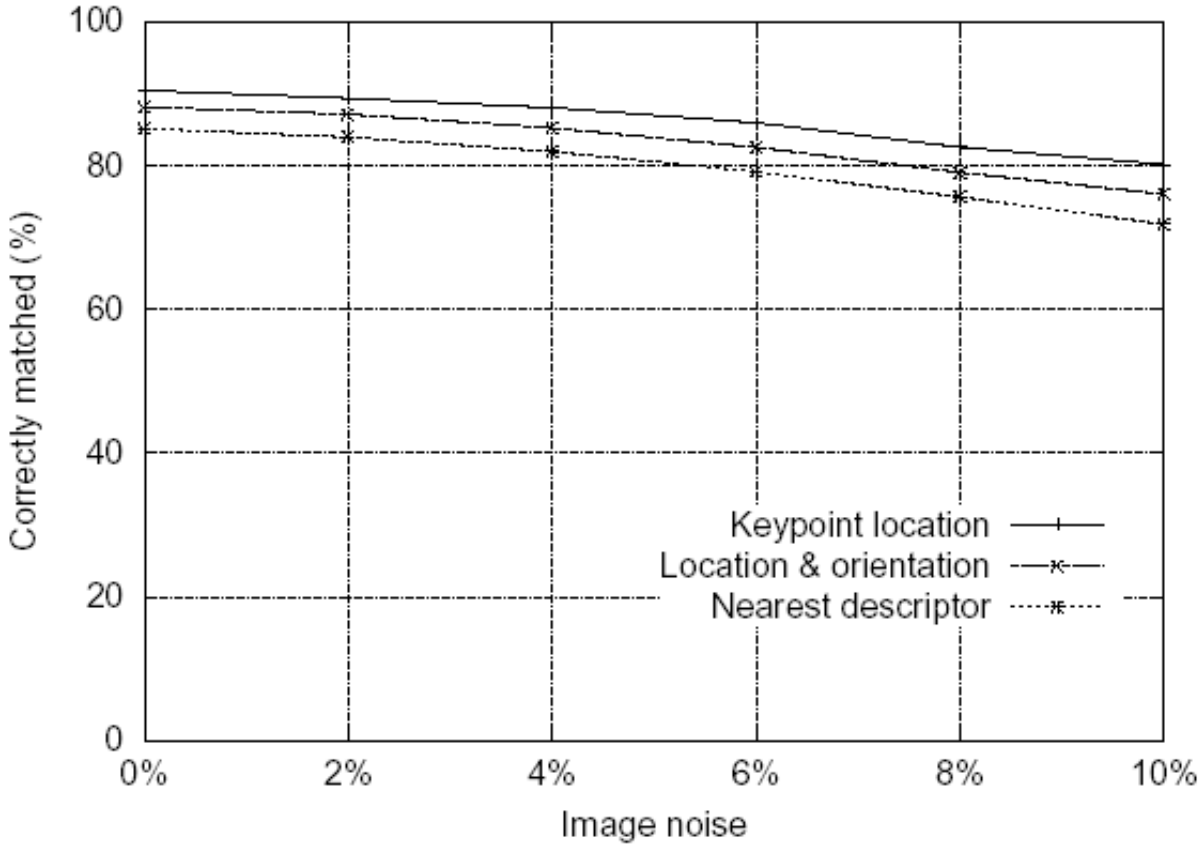


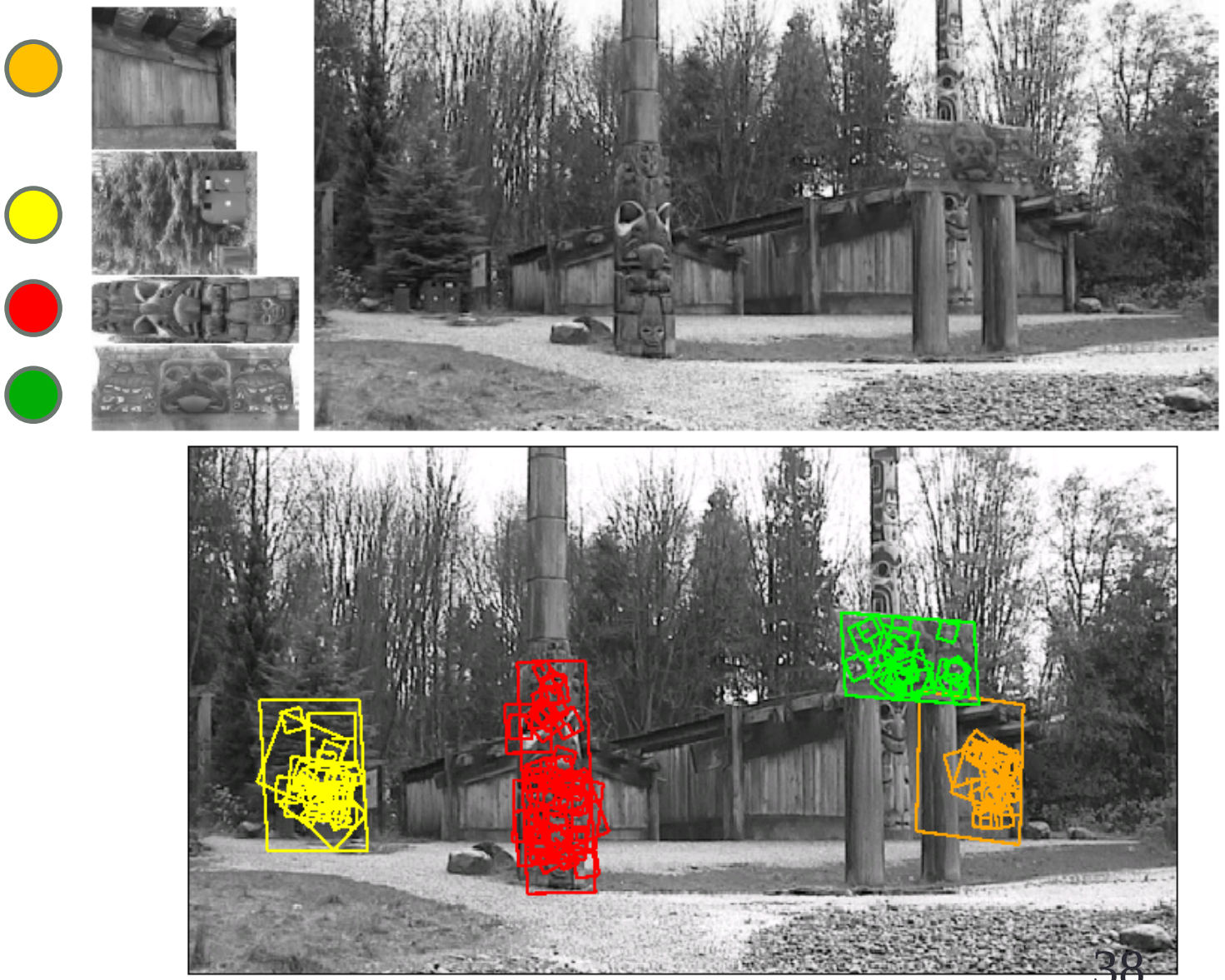
Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.

Feature stability to noise

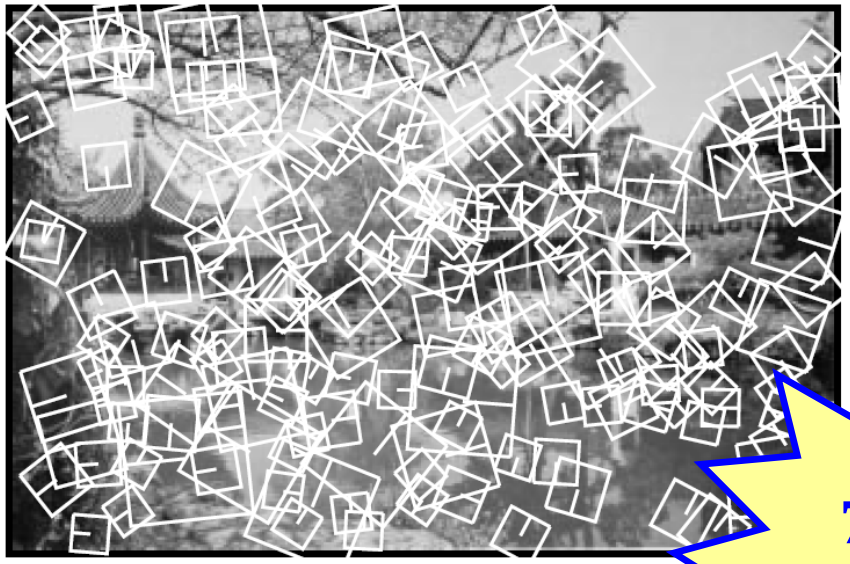
- Match features after random change in image scale & orientation, with differing levels of image noise
- Find nearest neighbor in database of 30,000 features



SIFT matching object pieces (for location)

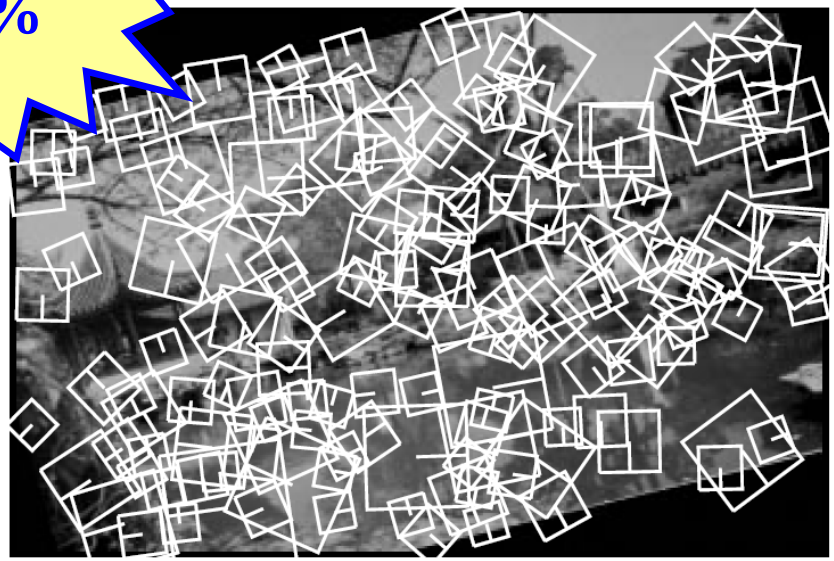


Experimental results



Original image

78%



Keypoints on image after rotation (15°), scaling (90%), horizontal stretching (110%), change of brightness (-10%) and contrast (90%), and addition of pixel noise

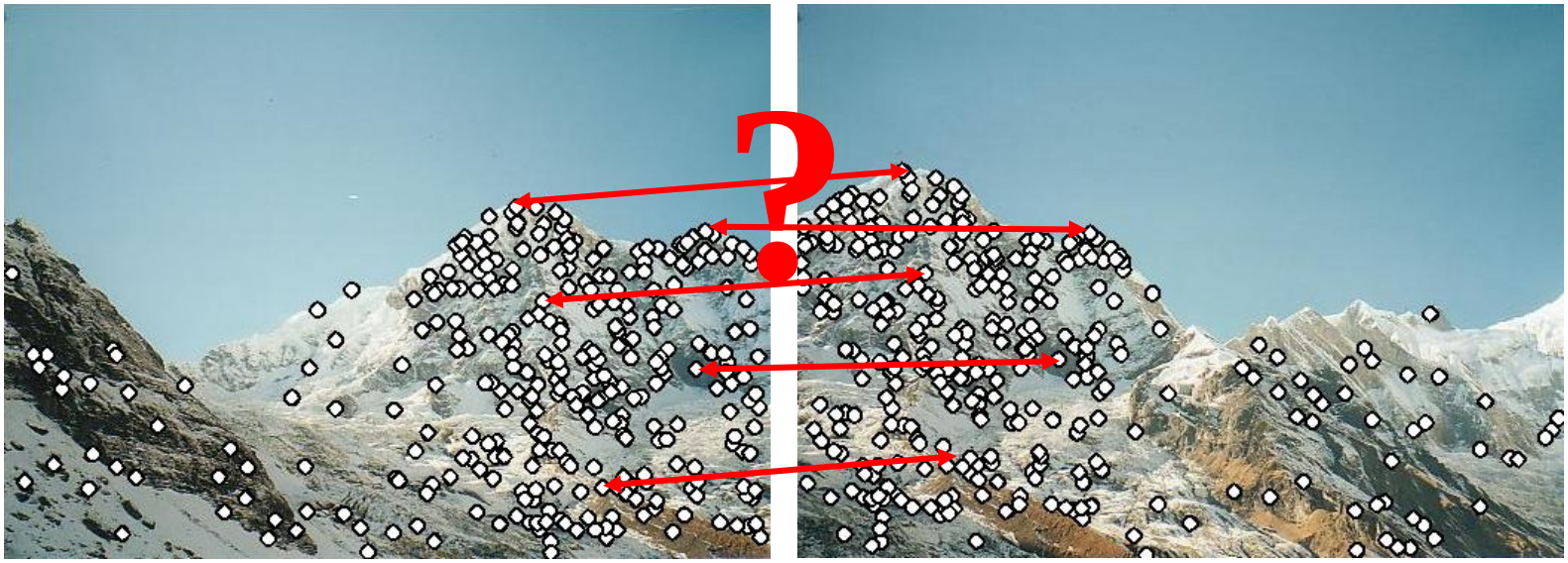
Experimental results (2)

Image transformation	Location and scale match	Orientation match
Decrease constrast by 1.2	89.0 %	86.6 %
Decrease intensity by 0.2	88.5 %	85.9 %
Rotate by 20°	85.4 %	81.0 %
Scale by 0.7	85.1 %	80.3 %
Stretch by 1.2	83.5 %	76.1 %
Stretch by 1.5	77.7 %	65.0 %
Add 10% pixel noise	90.3 %	88.4 %
All previous	78.6 %	71.8 %

20 different images, around 15,000 keypoints

Point Descriptors

- We know how to detect points
- We know how to describe them.
- Next question: How to match them?



Nearest-neighbor matching to feature database

- Could just do nearest neighbor. You will!
- Or, hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

Nearest neighbor techniques

- *k*-D tree and
- Best Bin First (BBF)

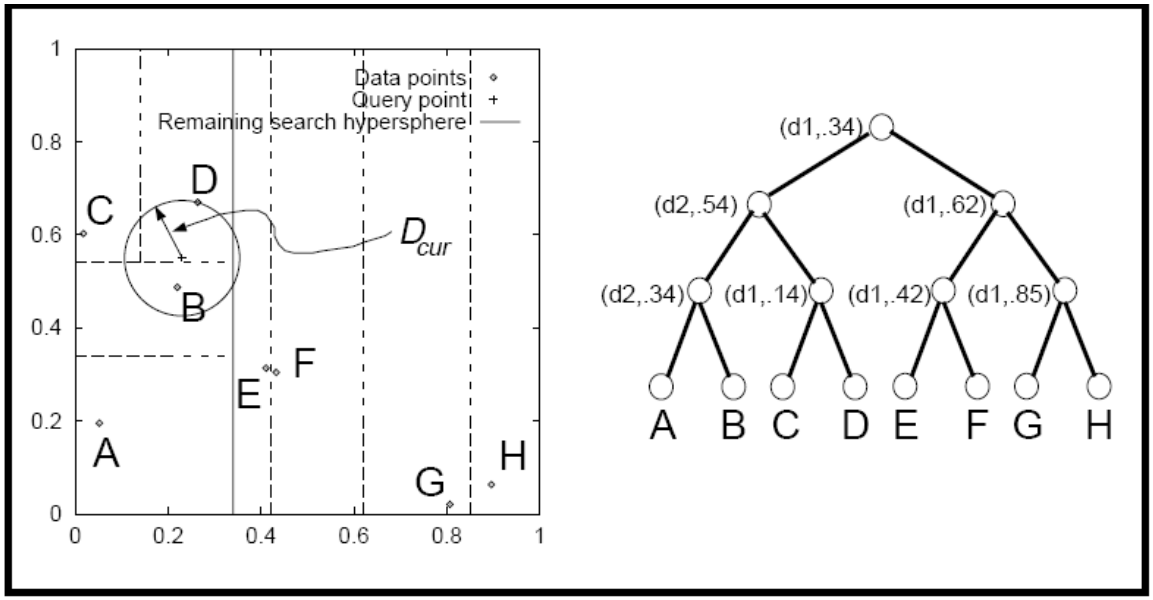
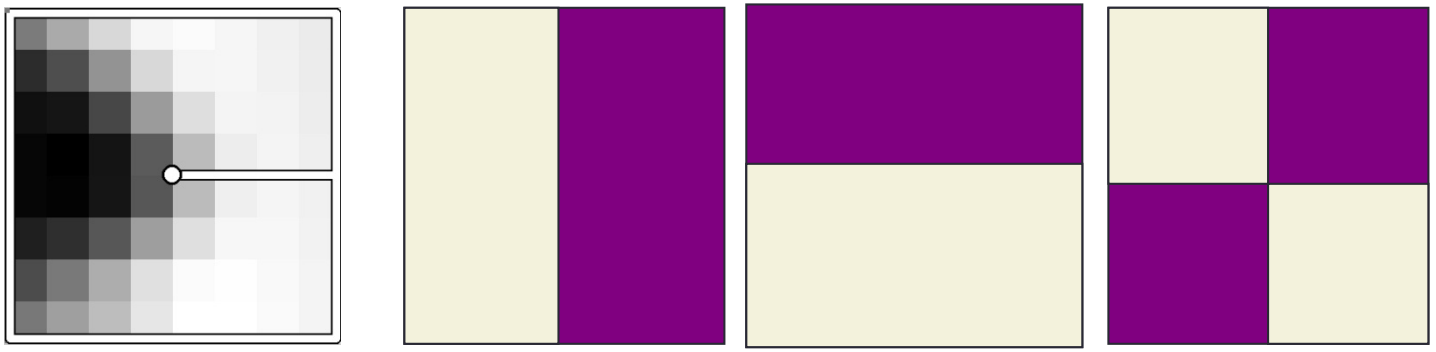


Figure 6: *k*d-tree with 8 data points labelled A-H, dimension of space k=2. On the right is the full tree, the leaf nodes containing the data points. Internal node information consists of the dimension of the cut plane and the value of the cut in that dimension. On the left is the 2D feature space carved into various sizes and shapes of bin, according to the distribution of the data points. The two representations are isomorphic. The situation shown on the left is after initial tree traversal to locate the bin for query point "+" (contains point D). In standard search, the closest nodes in the tree are examined first (starting at C). In BBF search, the closest bins to query point q are examined first (starting at B). The latter is more likely to maximize the overlap of (i) the hypersphere centered on q with radius D_{cur}, and (ii) the hyperrectangle of the bin to be searched. In this case, BBF search reduces the number of leaves to examine, since once point B is discovered, all other branches can be pruned.

Indexing Without Invariants in 3D Object Recognition, Beis and Lowe, PAMI'99

Wavelet-based hashing

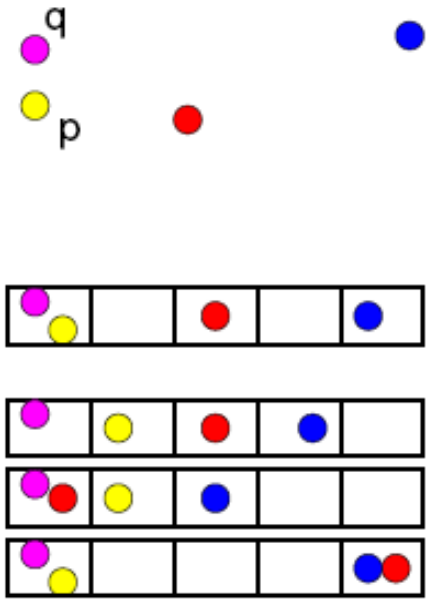
- Compute a short (3-vector) descriptor from an 8x8 patch using a Haar “wavelet”



- Quantize each value into 10 (overlapping) bins (10^3 total entries)
- [Brown, Szeliski, Winder, CVPR’2005]

Locality sensitive hashing

- Idea: construct hash functions $g: R^d \rightarrow U$ such that for any points p,q:
  - If $D(p,q) \leq r$, then $Pr[g(p)=g(q)]$ is ~~"high"~~ "not-so-small"
  - If $D(p,q) > cr$, then $Pr[g(p)=g(q)]$ is "small"
- Then we can solve the problem by hashing



Indyk and Motwani, 1998

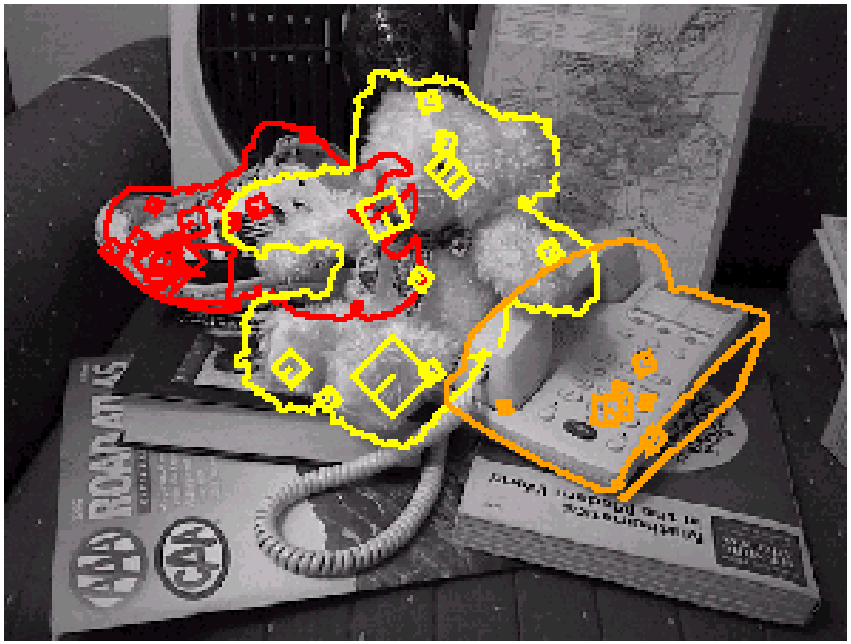
3D Object Recognition

- Extract outlines with background subtraction
- Compute keypoints
- Find possible matches.
- Search for consistent solution – such as affine.

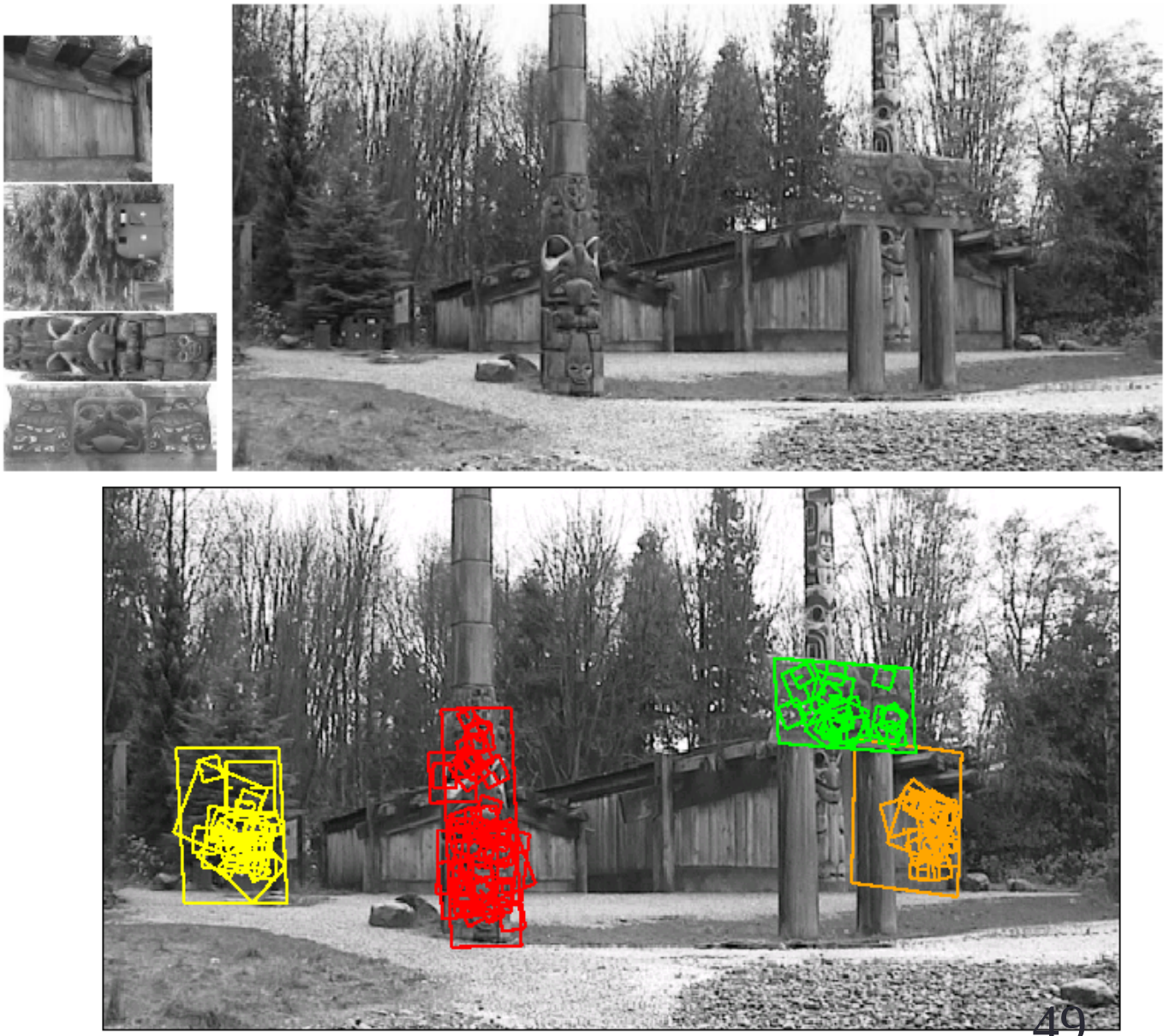
3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness

Recognition under occlusion



Location recognition



Sony Aibo (Evolution Robotics)

SIFT usage:

- Recognize charging station
- Communicate with visual cards